Quattor

M. Jouvin

Quattor

Michel JOUVIN
Laboratoire de l'Accélérateur Linéaire (LAL, UMR8607)
Domaine Universitaire
Bat. 200 - B.P. 34
91898 Orsay Cedex

Résumé

Quattor est un logiciel open-source qui permet d'automatiser l'installation initiale, la configuration et la gestion de serveurs utilisant Linux et « d'appliances ». Son originalité repose sur la description de la configuration désirée de l'ensemble des services d'une machine à l'aide d'un langage spécifiquement développé pour cet usage, Pan. Cette description est compilée et validée avant son déploiement par des modules de configuration qui s'exécutent sur les machines gérées et transforment la description en actions de configuration. Les fonctionnalités spécifiques du langage Pan ont rendu possible le développement d'une description générique pour chaque service qui permet une mutualisation entre sites de leur maintenance. Quattor autorise la mise en œuvre d'un grand nombre de modèles de site depuis un site isolé jusqu'à des sites distribués et de modèles d'administration depuis un modèle centralisé jusqu'à un modèle collaboratif ou la délégation de certaines fonctions.

1 Architecture

Quattor vise à permettre la gestion cohérente de l'ensemble des ressources d'un site. Un site est défini comme un ensemble arbitraire de ressources qui peuvent se trouver réparties dans plusieurs entités géographiques et offrir des services de nature différente (calcul, stockage, messagerie, web…) en utilisant des environnements logiciels (en particulier système d'exploitation) différents. Quattor a pour but de gérer efficacement aussi bien un grand nombre de configurations différentes qu'un grand nombre de machines concourant à l'implémentation d'un même service (par exemple, les worker nodes d'un cluster de calcul).

Pour cela, Quattor modélise un *site* comme en ensemble de groupe de *machines* dont la fonction est reflétée par une *personnalité*, elle-même définie comme un ensemble de *services*, chaque service pouvant lui-même se décomposer en services de plus bas niveau.

1.1 La description des configurations

D'une façon assez similaire à Puppet[2] (créé par une autre communauté à peu près en même temps…) et dans une certaine mesure cfengine[3] mais contrairement à beaucoup d'outils d'automatisation d'installation et gestion de clusters comme Chef[4] et Rocks[5], Quattor a développé un langage de description des configurations de type déclaratif, Pan, pour permettre de faire une description des configurations suivant ce modèle. Deux éléments importants pour implémenter cette approche sont :

- Une description de l'état souhaité pour chaque service et non pas du comment le configurer : c'est fonctionnalité est indispensable pour permettre de décrire de la même façon, et donc assurer la cohérence, l'utilisation d'un service dans des contextes différentes (par exemples des versions de système d'exploitation différentes). C'est aussi la seule façon de rendre la description indépendante de l'état actuel de la machine, condition indispensable d'une description qui par définition est statique et maintenue hors de la machine (on doit pouvoir décrire une machine pas encore installée).
- Une abstraction de la description par rapport aux détails de la configuration d'un service : c'est l'autre ingrédient indispensable pour permettre de rendre la description d'un service indépendant des détails de son implémentation et pouvoir ainsi cacher les évolutions internes de la mise en œuvre d'un service.

Par rapport au langage de Puppet, Pan offre deux propriétés intéressantes :

- C'est un langage compilé : cette compilation intervient avant le déploiement de la configuration sur la machine et se traduit par la génération d'un fichier XML contenant la configuration.
- La configuration doit obéir à un *schéma* qui définit la façon dont est organisée la configuration et les contraintes sur les différents éléments de la configuration (type, valeurs acceptables, cohérence entre plusieurs éléments de la

configuration…). La conformité de la description de la configuration d'une machine au schéma est validée lors de la compilation de la description : la non-conformité est une erreur fatale qui empêche le déploiement de la nouvelle configuration.

La conjugaison de ces deux propriétés du langage Pan permet d'implémenter une validation avant déploiement qui réduit le risque de déployer une configuration incorrecte et donc d'impacter le fonctionnement d'un service : une machine gérée par Quattor ne recevra une mise à jour de sa configuration (*profil*) que s'il a été validé. La validation peut porter sur tous les éléments de la configuration, y compris la présence d'un package dans les package repositories utilisés par la machine.

1.2 Le déploiement de la configuration

Après la description de la configuration qui se fait sans aucune interaction avec les machines gérées, c'est la phase qui se déroule sur ces dernières. Une machine gérée par Quattor est notifiée lorsqu'une nouvelle configuration (*profile*) est disponible. Comme indiqué précédemment cette description, sous la forme d'un fichier XML, est totalement indépendante de l'état de la machine et contient tous les aspects de la configuration, y compris les packages à déployer.

La traduction d'une configuration en action de configuration permettant de passer de l'état actuel d'une machine à son nouvel état est faite par des agents appelés *configuration modules*. Ces agents fonctionnent comme des plugins en charge de la configuration d'un service particulier. Il en existe un grand nombre couvrant à la fois les services les plus courants d'un système Linux, ainsi que tous les services spécifiques permettant de gérer le middleware de grille gLite (historiquement le premier objectif de Quattor).

Ces modules de configuration sont actuellement des scripts Perl de complexité variable suivant les services. Ils utilisent l'API standard fournit par Quattor pour accéder et manipuler la description de la configuration. Un travail est actuellement en cours pour permettre l'utilisation d'autres langages pour l'écriture de ces modules, en particulier Python.

1.3 Installation initiale

Une des spécificités de Quattor est de permettre d'utiliser la description de la configuration de la machine pour en faire l'installation initiale, afin d'assurer la cohérence entre la configuration initiale d'une machine et celle désirée (décrite dans Quattor) et d'éviter ainsi une duplication d'effort pour maintenir un système d'installation, même minimal, séparé.

Cette pour cette raison en particulier que la description d'une machine dans Quattor inclut une description minimale de son hardware. Dans Quattor, la description de la configuration contient à la fois l'information sur les interfaces réseaux utilisés, sur la version d'OS à déployer, sur les disques, leur partitionnement et les file systems qui les utilisent. Ceci permet, par l'exécution d'une commande spécifique, de configurer l'environnement DHCP et TFTP nécessaire pour une installation à travers le réseau (PXE) et de générer le fichier de configuration Kickstart (actuellement Quattor ne gère que des distributions de type Red Hat) pour créer un environnement de base (partitions disques, langue, version d'OS…) conforme à la configuration souhaitée et permettant l'installation de l'agent Quattor. Une fois l'environnement de base installé, il s'exécutera pour implémenter la configuration décrite suivant le processus standard de déploiement de configuration.

1.4 La base de données de configuration

Intermédiaire entre l'administrateur système qui va maintenir la description des configurations et les machines gérées par Quattor, il y a au cœur de Quattor une base de données de configuration. C'est elle qui va fournir l'interface d'utilisation de Quattor permettant de modifier, valider et déployer une configuration. Une des fonctionnalités essentielles de la base de données de configuration est d'assurer un versionning des changements de configuration pour permettre un retour en arrière simple sur toute modification.

Il existe actuellement deux implémentations différentes de bases de données de configuration (voir ci-dessous), offrant des fonctionnalités et une interface utilisateur différents. Mais ces deux implémentations utilisent le même langage de description Pan, les mêmes modules pour le déploiement des configurations et partagent les mêmes éléments de description générique de services (*templates*).

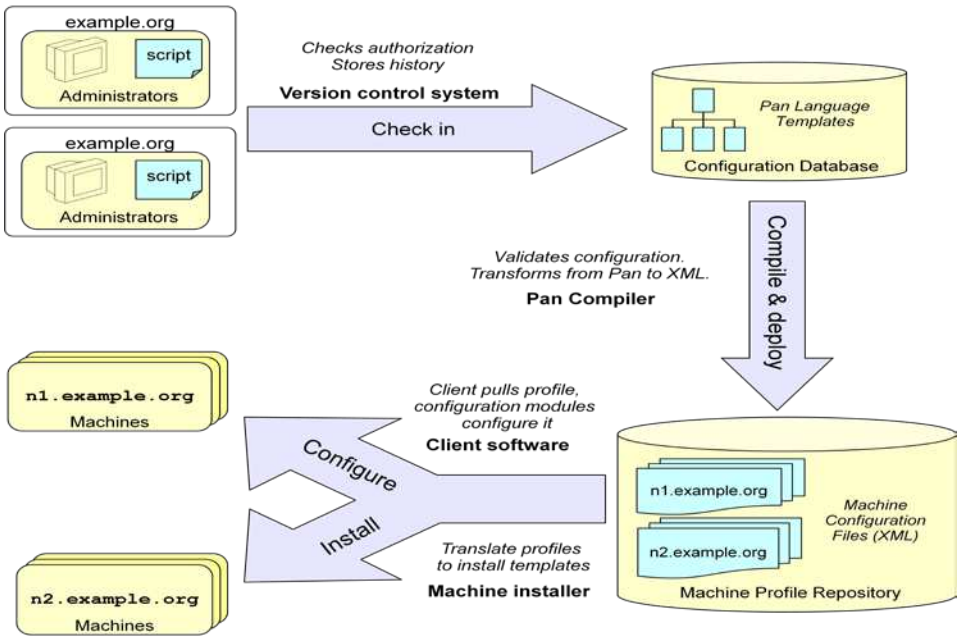

Figure 1 Quattor Workflow

2 Les différents composants Quattor

2.1 Les bases de données de configuration

Comme indiqué ci-dessus, la base de données de configuration est l'élément central de Quattor et fournit l'interface d'utilisation de Quattor : ce composant est à la fois en charge du stockage des descriptions de configuration (*templates* Pan), de leur versionning et de leur compilation. C'est donc la « face visible » de Quattor pour un administrateur système. C'est aussi le seul élément de Quattor dont il peut exister plusieurs implémentations sans impact sur le modèle général, la description des configurations et leur déploiement.

L'implémentation de la base de données de configuration la plus utilisée actuellement est SCDB. Basée sur SVN, elle a rapidement remplacé la première implémentation basée sur CVS (CDB). Prenant en compte la montée en puissance des postes de travail, SCDB favorise l'utilisation de l'environnement de travail de l'utilisateur pour éviter le goulet d'étranglement d'un serveur unique : seule la phase finale du déploiement intervient sur un serveur central pour en assurer la sérialisation (deux déploiements en parallèle donneraient des résultats imprévisibles). SCDB étant écrit en Java (de même que le compilateur Pan), il est possible de l'utiliser sur n'importe quelle plateforme. Les outils SCDB sont facilement intégrables dans l'IDE Eclipse pour faciliter le développement des *templates* (mais ce n'est pas un pré-requis !).

Plus récemment, une nouvelle implémentation de la base de données de configuration a vue le jour, Aquilon. Son architecture est basée sur un *broker* qui a la responsabilité d'exécuter les actions, à la différence de SCDB qui utilise le poste de travail de l'administrateur système. Aquilon utilise Git comme outil de versionning. Le design d'Aquilon a été fait avec comme objectif de gérer un très grand nombre de machines : le site qui l'a développé gère actuellement 50 000 machines réparties dans le monde avec une seule instance de Aquilon ! Dans Aquilon, une machine est modélisée comme l'association d'un hardware, d'une configuration software appelée *personnalité*, et de paramètres spécifiques comme les paramètres réseaux qui peuvent être regroupés dans des ensembles appelés *archetypes*. Contrairement à SCDB, dans lequels toute la description de la configuration est contenue dans des *templates* Pan, Aquilon utilise les *templates* Pan comme une bibliothèque pour implémenter les personnalités et les *archetypes*. Mais leur association dans une machine donnée et les paramètres spécifiques à la machine sont maintenus dans une base de donnée relationnelle (SQLite, Postgres ou Oracle actuellement). Les informations de cette base de données sont utilisées pour générer dynamiquement les *templates* Quattor reflétant cette information au moment du déploiement, ce qui simplifie considérablement l'utilisation de Quattor pour les opérations courantes. L'utilisation interne de Git a permis d'implémenter une fonctionnalité de *sandboxing*, en utilisant les fonctionnalités de clonage Git, pour simplifier le développement et le test de nouvelles versions des *templates* Pan sans impact sur la production.

2.2 QWG Templates

Du fait de fonctionnalités spécifiques au langage Pan (en particulier, la définition de valeur par défaut pour chaque élément de la configuration), il est possible de développer une description générique des services qui peut être utilisées telles quelles par un site sans besoin de les modifier pour l'adapter à sa configuration. Ces templates standards se comportent comme une bibliothèque de description de configuration qui est instanciée dans le contexte de chaque machine pour refléter sa configuration. Cette fonctionnalité unique permet à la communauté des sites Quattor de partager l'effort de développement et de maintenance de la description des services : chaque fois qu'un site ajoute une nouvelle option de configuration dans ces templates standards elle devient disponible pour les autres sites par une simple opération de mise à jour, avec une garantie de compatibilité avec les versions antérieures (testée lors du processus de développement) donc sans impact sur la configuration spécifique du site.

Ces templates standards, connus sous le nom de QWG templates, couvrent plusieurs besoins différents et sont organisés en plusieurs sous-ensembles :

- Core templates : ils couvrent l'installation et la configuration du système exploitation, des services de base (DNS, NIS, LDAP, NFS…) et de services génériques comme SMTP ou HTTP. Les core templates incluent aussi la possibilité de configurer Nagios pour le monitoring du site à partir de la description des ressources faites dans Quattor. Au niveau du système d'exploitation, Quattor ne sait actuellement déployer et gérer que les distributions Red Hat (RHEL et Fedora) et leurs dérivées (CentOS, Scientific Linux).
- Middleware gLite[6] : la gestion de sites grille étant à l'origine de Quattor, un des sous-ensembles de templates standards permet de gérer l'ensemble des services grille d'un site EGI[7] ou utilisant le middleware gLite.
- StratusLab : StratusLab[8] est un projet européen dont l'objectif est de développer une plateforme open-source pour le déploiement de cloud IaaS. En plus du logiciel de la plateforme, le projet StratusLab maintient la description générique des services StratusLab, utilisable par Quattor pour déployer un site StratusLab.

Les sous-ensembles gLite et StratusLab s'appuient sur les core templates pour la configuration des services fournis par le système d'exploitation.

Actuellement la quasi-totalité des sites Quattor utilisent ces templates standards, au moins les core templates, pour construire la description de leurs ressources.

2.3 QuattorFS

Ce composant, installé sur les machines gérées par Quattor, permet d'accéder la configuration Quattor et de naviguer à l'intérieur en la rendant visible comme un file system, comme c'est le cas pour la configuration du système avec /proc. L'implémentation repose sur un file system FUSE.

2.4 Quattor Remote Deployer

Ce composant, QRD, développé récemment, permet de gérer des appliances ou toute ressource sur laquelle on ne peut pas installer l'agent de déploiement Quattor mais qui dispose d'une capacité de gestion à distance (API spécifique, ssh…). Il implémente un *proxy* qui reçoit les notifications lors des changements de configuration (telle que décrite dans Quattor) des appliances et utilise la méthode de connexion appropriée pour implémenter les changements à travers l'utilisation de modules de configuration (fonctionnellement identiques à ceux d'une machine Linux) spécifiques à l'appliance. Pour chaque type d'appliance, Quattor définit un schéma spécifique adapté (un switch réseau ne va pas être décrit de la même façon qu'une machine Linux).

QRD est actuellement utilisé pour gérer des clusters VMware ESX et des switches réseau CISCO.

3 Quelques exemples d'utilisation

3.1 Gestion de site de la grille EGI

La grille européenne de production EGI, à travers les différents projets (EDG, EGEE) qui ont permis son développement depuis le début des années 2000, est le cadre dans lequel Quattor a été conçu et initialement développé. Le but était de gérer de façon cohérente et reproductible des sites de grande taille.

L'utilisation par tous les sites de la grille d'un middleware commun (gLite) a conduit au développement des QWG templates mentionnés précédemment. Le but était de fournir une description de la configuration des différents services grille qui puissent être partagée entre sites, facilitant la mise en œuvre des services dans les sites ayant peu de personnes pour l'administration du site ou

bien peu d'expertise dans les services grille. Cette description est maintenue par quelques volontaires pour prendre en compte l'évolution des services.

L'utilisation de Quattor comme outil de gestion de la configuration des services grille reste dominante dans la communauté Quattor, même si récemment des sites sans rapport avec la grille de production l'ont adopté. Cette capacité de mutualisation de l'effort entre sites lorsqu'ils mettent en oeuvre les mêmes services est généralement une motivation déterminante du choix de Quattor.

3.2 Gestion d'images virtuelles

Il s'agit d'une évolution de l'utilisation de Quattor, dans un contexte de ressources virtualisées à grande échelle (par exemple ressources de type cloud). Dans ce type d'environnement, il y a généralement plusieurs machines virtuelles qui instancient le même service (par exemple un serveur web) avec des images de machine virtuelle identiques. L'approche classique dans Quattor d'un profil par machine n'est pas très appropriée car elle conduit à gérer potentiellement un grand nombre de profils identiques.

Les expériences de gestion de ressources virtualisées montrent que la création initiales et surtout la maintenance des images utilisées par les machines virtuelles est l'un des principaux défis de la virtualisation tant pour la consolidation de service que pour les clouds (tous les utilisateurs n'ont pas la compétence pour produire et maintenir des images virtuelles). Non maitrisé ou géré au coup par coup, ce défi peut conduire à des difficultés opérationnelles majeures : incohérences de l'environnement entre plusieurs images, difficulté à déployer les correctifs de failles de sécurité… Ces difficultés annuleraient rapidement les bénéfices opérationnels de la virtualisation.

Dans le cadre du projet StratusLab, une nouvelle approche a été proposée et implémentée : l'utilisation de Quattor pour construire et gérer les images de machines virtuelles plutôt que les machines virtuelles elle-même. Comme indiqué précédemment, en plus du logiciel de la plateforme, StratusLab fournit l'intégration avec Quattor comme outil de déploiement de la plate-forme (description générique des services et modules de configuration spécifiques). Cette approche est mise en œuvre en particulier au LAL dans le cadre d'un cloud dédié à la consolidation de serveurs [8].

3.3 Des exemples de site

Les trois sites suivants illustrent les possibilités de Quattor de mettre en œuvre des modèles de site et d'administration très différents :

- CERN [9] : fortement impliqué dans le design et le développement initial de Quattor, le CERN est le premier site à avoir déployé Quattor en production. Depuis toujours un « grand » site par son nombre de ressource, il gère environ 10 000 machines avec Quattor correspondant à beaucoup de services de nature différente. Le CERN a développé tout un écosystème d'outils spécifiques pour la gestion du site et la gestion du cycle de vie des machines autour de Quattor.
- GRIF [10] : c'est un des grands sites de la grille de production européenne EGI et en particulier de sa composante WLCG[11]. Situé en région parisienne, il est une initiative conjointe de 6 laboratoires (CNRS/IN2P3 dont le LAL, CEA/Irfu) et met en œuvre un site unique avec des ressources et une équipe technique (15 personnes impliquées) réparties sur les 6 sites. GRIF représente aujourd'hui 1500 machines (10 000 cœurs de calcul et des serveurs de disque offrant 2PB consolidés). Quattor a été un outil majeur pour permettre la mise en place de ce site distribué et plusieurs des laboratoires impliqués dans GRIF utilisent aussi Quattor pour gérer leurs machines internes (non grille). Plusieurs de ses membres contribuent activement au développement de Quattor.
- Morgan Stanley : Grande banque, Morgan Stanley a choisi Quattor il y a 3 ans pour remplacer son système maison précédent, après une étude approfondie des différentes solutions open-source disponibles. Très engagé dans l'open-source, MS contribue activement au développement de Quattor et est à l'origine de Aquilon.

4 La communauté et la documentation

La communauté Quattor représente une cinquantaine de site, principalement situés en Europe mais aussi en Asie (Chine). Une grande partie de ces sites est lié à la grille de production EGI mais plusieurs sites utilisent Quattor pour gérer l'intégralité de leur site et non pas seulement leurs ressources grille. Elle se réunit deux fois par an (Quattor Workshop).

La communauté Quattor est organisée autour d'un wiki, accessible à partir du portail Quattor [1]. Le code source est hébergé par SourceForge [12] et peut être téléchargé par toute personne intéressée. Il utilise une license Apache2. Le support et la discussion entre les membres de la communauté se fait à travers des mailing lists et une conference room Jabber (voir information sur le wiki).

Le développement de Quattor est assuré par les sites qui l'utilisent, assurant ainsi son adaptation aux besoins réels des sites et un test permanent des nouvelles fonctionnalités développées. Tous les sites sont encouragés à contribuer en fonction de leur expertise et de leurs moyens mais le processus de développement est coordonné par un groupe de personne volontaire à travers un

processus de développement « agile », basé sur la méthodologie SCRUM [15]. L'état des développements en cours est publié en permanence sur le wiki de la communauté Quattor.

5 Conclusion

Quattor reste un outil original dans l'écosystème des outils de gestion de configuration dont les principales caractéristiques sont :

- Une description de l'**état final** d'une machine, centrée sur les services
- Une validation de la description des configurations **avant** leur déploiement
- Une gestion de l'installation initiale **et** de l'évolution de la configuration à partir de la même description
- Le **versionning** des configurations et un retour en arrière simple sur tout changement, y compris le downgrade de packages.
- La **reproductibilité** des installations
- Un **partage entre sites** de la maintenance des descriptions de configuration pour les services utilisés par plusieurs sites
- Une **extension** facile pour la gestion de nouveaux services
- La capacité de gérer des « **appliances** »

La communauté d'origine, liée à la grille de production européenne, s'est élargie ces dernières années à des partenaires non académiques. Dans le cadre du projet StratusLab, Quattor a évolué pour pouvoir être utilisé comme un gestionnaire d'image virtuelle, en plus de la gestion de machine physique.

Le développement de Quattor est pris en charge par la communauté des sites qui l'utilisent, assurant ainsi son adéquation aux besoins réels des administrateurs système. La diversité des sites présents permet la prise en compte différents modèles d'administration de site. Utilisant une méthode de développement Agile [13], le développement de Quattor est basé sur une interaction rapide avec ses utilisateurs et les priorités définies par la communauté, dans l'esprit du mouvement DevOps [14] visant à réduire la fracture entre les développeurs de logiciel et ceux qui le mettent en oeuvre.

Références

[1] http://quattor.org

[2] http://puppetlabs.com/

[3] http://cfengine.com/

[4] http://www.opscode.com/chef/

[5] http://www.rocksclusters.org/wordpress/

[6] http://www.eu-emi.eu/ et http://glite.org

[7] European Grid Infrastructure, http://www.egi.eu/

[8] http://stratuslab.eu, voir aussi JRES 2011, Virtualisation de serveurs à l'aide d'un cloud, Guillaume Philippon

[9] http://cern.ch

[10] Grille au service de la Recherche en Ile-de-France, http://grif.fr

[11] Worldwide LHC Computing Grid, http://lcg.web.cern.ch/lcg/

[12] Projet Quattor @SourceForge : http://sourceforge.net/projects/quattor

[13] Agile Manifesto, http://agilemanifesto.org/

[14] http://en.wikipedia.org/wiki/DevOps

[15] http://en.wikipedia.org/wiki/Scrum_(development)